PHPillow & CouchDB

Kore Nordmann <kore@php.net>

July 11, 2009

About me

- Kore Nordmann, <kore@php.net>, <kn@ez.no>
- Long time PHP developer
- Regular speaker, author, etc.
- Studies computer science in Dortmund
- Active open source developer:
  - eZ Components (Graph, WebDav, Document), Arbit, PHPUnit, Torii, *PHPillow*, KaForkL, Image 3D, WCV, ...

Outline

Databases

- Who uses an RDBMS (relational database management system)?

Databases

- Who uses an RDBMS (relational database management system)?
- Who uses a hash based “database”? (MemcacheDB, ...)

Databases

- Who uses an RDBMS (relational database management system)?
- Who uses a hash based “database”? (MemcacheDB, ...)
- Who uses a “document” based database?
  - Amazon S3 (SimpleDB)
  - StrokeDB (Ruby)
  - FeatherDB (Java port of CouchDB)
  - CouchDB

Outline

PHPillow

- A PHP based object oriented client for CouchDB

h more comfortable

PHPillow

- A PHP based object oriented client for CouchDB
- The Pillow to make using the couch more comfortable

CouchDB vs. RDBMS

- RDBMS: Static data with dynamic views
- CouchDB: Dynamic data with “static” views

CouchDB

- Document based database

CouchDB

- Document based database
  - Schema-less

CouchDB

- Document based database
  - Schema-less
  - Store anything you want, even deep structures

- Document based database
  - Schema-less
  - Store anything you want, even deep structures
  - Arbitrary JSON objects, attachments

CouchDB

- Document based database
  - Schema-less
  - Store anything you want, even deep structures
  - Arbitrary JSON objects, attachments
  - No inter-document consistency

CouchDB

- Document based database
  - Schema-less
  - Store anything you want, even deep structures
  - Arbitrary JSON objects, attachments
  - No inter-document consistency
- Apache Project

CouchDB

- Document based database
  - Schema-less
  - Store anything you want, even deep structures
  - Arbitrary JSON objects, attachments
  - No inter-document consistency
- Apache Project
- ACID compliant

CouchDB

- Document based database
  - Schema-less
  - Store anything you want, even deep structures
  - Arbitrary JSON objects, attachments
  - No inter-document consistency
- Apache Project
- ACID compliant
- MVCC (Multiversion Concurrency Control)

CouchDB

- Document based database
  - Schema-less
  - Store anything you want, even deep structures
  - Arbitrary JSON objects, attachments
  - No inter-document consistency
- Apache Project
- ACID compliant
- MVCC (Multiversion Concurrency Control)
- Trivial replication

CouchDB

- Document based database
  - Schema-less
  - Store anything you want, even deep structures
  - Arbitrary JSON objects, attachments
  - No inter-document consistency
- Apache Project
- ACID compliant
- MVCC (Multiversion Concurrency Control)
- Trivial replication
- Communicates with clients using HTTP

Architecture

- Written in/for Erlang/OTP
  - Rocks! ... why?

Architecture

- Written in/for Erlang/OTP
  - Rocks! ... why?
- Scales nearly linearly with amount of processors

Architecture

- Written in/for Erlang/OTP
  - Rocks! ... why?
- Scales nearly linearly with amount of processors
- Highly fault tolerant (9 nines @Ericsson)
  - Live updates of application code

Architecture

- Written in/for Erlang/OTP
  - Rocks! ... why?
- Scales nearly linearly with amount of processors
- Highly fault tolerant (9 nines @Ericsson)
  - Live updates of application code
- Developed by Ericsson for telephony systems

ACID compliant

- ACID = atomicity, consistency, isolation and durability
  - The key-constraint for databases

ACID compliant

- ACID = atomicity, consistency, isolation and durability
  - The key-constraint for databases
  - Requires fsync() calls

ACID compliant

- ACID = atomicity, consistency, isolation and durability
  - The key-constraint for databases
  - Requires fsync() calls
  - fsync() calls are THE speed limit for databases [Leh09]

ACID compliant

- ACID = atomicity, consistency, isolation and durability
  - The key-constraint for databases
  - Requires fsync() calls
  - fsync() calls are THE speed limit for databases [Leh09]
- Optionally disable fsync() calls, loosing guaranteed storage
  - Major speed up
  - Useful for bulk insertions

- MVCC (Multiversion Concurrency Control)
  - Multiple revisions of documents in the database
    - Identified using some revision-ID.

- MVCC (Multiversion Concurrency Control)
  - Multiple revisions of documents in the database
    - Identified using some revision-ID.
  - Updates and deletes operate on specific revisions

- MVCC (Multiversion Concurrency Control)
  - Multiple revisions of documents in the database
    - Identified using some revision-ID.
  - Updates and deletes operate on specific revisions
  - Provides a consistent database to multiple clients

- MVCC (Multiversion Concurrency Control)
  - Multiple revisions of documents in the database
    - Identified using some revision-ID.
  - Updates and deletes operate on specific revisions
  - Provides a consistent database to multiple clients
    - Even on concurrent updates.

- MVCC (Multiversion Concurrency Control)
  - Multiple revisions of documents in the database
    - Identified using some revision-ID.
  - Updates and deletes operate on specific revisions
  - Provides a consistent database to multiple clients
    - Even on concurrent updates.
    - Conflict handling left to the clients

Replication

- Trivial incremental user-requested pull- or push-replication [Apa09]

Replication

- Trivial incremental user-requested pull- or push-replication [Apa09]
- Just call `replicate(source, target)`

Replication

- Trivial incremental user-requested pull- or push-replication [Apa09]
- Just call `replicate(source, target)`
  - Pull: local-source $\longrightarrow$ remote-target

Replication

- Trivial incremental user-requested pull- or push-replication [Apa09]
- Just call `replicate(source, target)`
  - Pull: local-source $\longrightarrow$ remote-target
  - Push: remote-source $\longrightarrow$ local-target

Replication

- Trivial incremental user-requested pull- or push-replication [Apa09]
- Just call `replicate(source, target)`
  - Pull: local-source → remote-target
  - Push: remote-source → local-target
- Everything else happens automatically

Replication

- Trivial incremental user-requested pull- or push-replication [Apa09]
- Just call `replicate(source, target)`
  - Pull: local-source → remote-target
  - Push: remote-source → local-target
- Everything else happens automatically
  - Conflicts in multi-write setups are resolved deterministically

Communicates with HTTP (1/3)


```
1 $ curl -I -X PUT http://localhost:5984/phpug
2 HTTP/1.1 201 Created
3 Server: CouchDB/0.10.0a773833 (Erlang OTP/R12B)
4 Content-Type: text/plain;charset=utf-8
5
6 {"ok":true}
```

Communicates with HTTP (2/3)

```
$ curl -I -X GET http://localhost:5984/phpug
HTTP/1.1 200 OK
Server: CouchDB/0.10.0a773833 (Erlang OTP/R12B)
Content-Type: text/plain;charset=utf-8

{
    "db_name": "phpug",
    "doc_count": 0,
    "doc_del_count": 0,
    "update_seq": 0,
    "purge_seq": 0,
    "compact_running": false,
    "disk_size": 4096,
    "instance_start_time": "1243673303593969",
    "disk_format_version": 2
}
```

Communicates with HTTP (3/3)

```
1 $ curl -I -X GET http://localhost:5984/unknown
2 HTTP/1.1 404 Object Not Found
3 Server: CouchDB/0.10.0a773833 (Erlang OTP/R12B)
4 Content-Type: text/plain;charset=utf-8
5
6 {"error":"not_found","reason":"Missing"}
```

Views

- Map-reduce implementations for “static” views

```
return array_reduce(
    array_map(
        function ( $character )
        {
            return ord( $character );
        },
        str_split( $string, 1 )
    ),
    function ( $sum, $character )
    {
        return $sum + $character;
    }
) % 16;
```

Views

- Map-reduce live session

Outline

PHPillow

- Object-oriented client for CouchDB
- PHP >= 5.2 since last release (5.3 only before)
- >96% test coverage

PHPillow

- Object-oriented client for CouchDB
- PHP >= 5.2 since last release (5.3 only before)
- >96% test coverage
- Still in alpha state

PHPillow

- Object-oriented client for CouchDB
- PHP >= 5.2 since last release (5.3 only before)
- >96% test coverage
- Still in alpha state
  - Since CouchDB is still “alpha”

Document examples

- Document creation example

```php
// Create a document
$doc = new phpillowUserDocument();
$doc->login = 'kore';
$doc->name  = 'Kore Nordmann';
$doc->data  = array(
    'mail' => "kore@php.net",
    // ...
);
$id = $doc->save();

// Fetch a document by ID
$doc = new phpillowUserDocument( $id );
```

Outline

View examples

- Issue tracker views from arbit
  - *The* next generation issue tracking system ;)

Simple view

- Show all user documents

```
1  function( doc ) {
2    if ( doc.type == "user" ) {
3      emit( null, doc._id );
4    }
5  }
```

Simple view

- Show only unregistered users

```
function( doc ) {
  if ( doc.type == "user" &&
     doc.valid !== "0" &&
     doc.valid !== "1" ) {
    emit( doc.valid, doc._id );
  }
}
```

Simple view

- Emit values from deep structures

```
1  function( doc ) {
2    if ( doc.type == "group" ) {
3      for ( var i = 0; i < doc.users.length; ++i ) {
4        emit( doc.users[i], doc.permissions );
5      }
6    }
7  }
```

Map-Reduce: Full-Text-Search

- Index all documents by all their words

```
1  function( doc ) {
2    if ( doc.type == "tracker_issue" ) {
3      // Simple word indexing, does not respect overall
           occurences of words,
4      // stopwords, different word seperation characters,
           or word variations.
5      var text  = doc.title.replace( /[\s:.,!?-]+/g, " " )
           +
6          doc.text.replace( /[\s:.,!?-]+/g, " " );
7      var words = text.split( " " );
8      for ( var i = 0; i < words.length; ++i ) {
9        value = {};
10       value[doc._id] = 1;
11       emit( words[i].toLowerCase(), value );
12     }
13   }
14 }
```

Map-Reduce: Full-Text-Search

- Reduce by word count

```
1  function( keys, values ) {
2    var count = {};
3    for ( var i in values ) {
4      for ( var id in values[i] ) {
5        if ( count[id] ) {
6          count[id] = values[i][id] + count[id];
7        } else {
8          count[id] = values[i][id];
9        }
10     }
11   }
12   return count;
13 }
```

URLs

- Apache CouchDB: `http://couchdb.org/`
- PHPillow: `http://kore-nordmann.de/projects/phpillow/`
- CouchDB use case: `http://kore-nordmann.de/blog/phpillow_php_couchdb_wrapper.html`
  - Implementing a user permission system using CouchDB

The end

- Open questions?
- Further remarks?
- Contact
  - Mail: <kore@php.net>
  - Web: `http://kore-nordmann.de/` (Slides will be available here soonish)
  - Twitter: `http://twitter.com/koredn`

Bibliography I

- Apache, *Replication*, http://wiki.apache.org/couchdb/Replication, July 2009.
- Jan Lehnardt, *Caveats of evaluating databases*, http://jan.prima.de/~jan/plok/archives/176-Caveats-of-Evaluating-Databases.html, June 2009.